Jrystal: A JAX-based Differentiable Density Functional Theory Framework for Materials

Tianbo Li[1,a], **Zekun Shi**[1,2,b], **Stephen Gregory Dale**[3,c], **Giovanni Vignale**[3,d], **Min Lin**[1,e]
[1]SEA AI Lab
[2]School of Computing, National University of Singapore
[3]Institute for Functional Intelligent Materials, National University of Singapore
{[a]litb, [b]shizk, [e]linmin}@sea.com,
{[c]sdale, [d]vgnl.g}@nus.edu.sg

Abstract

Density functional theory (DFT) is crucial for studying materials at the atomic level, known for its accurate predictions and computational efficiency. However, integrating AI techniques into current DFT packages is challenging due to the iterative nature of the self-consistent field methods. To address these challenges, we developed *Jrystal*, a JAX-based differentiable DFT framework that integrates modern deep learning frameworks with automatic differentiation and adopts a direct optimization approach using plane wave bases. This makes *Jrystal* a powerful tool for designing new DFT algorithms with machine learning techniques.

1 Introduction

Density functional theory (DFT) [9] is a key method for studying the behavior of materials at the atomic level, gaining widespread use in both scientific research and industry [12]. DFT allows scientists to study the electronic structure of materials by focusing on the density of electrons rather than solving the more complex many-body problem, making it possible to predict important properties of materials—such as how they conduct electricity, respond to magnetic fields, or transfer heat—with high accuracy, while still being computationally manageable. Despite requiring some approximations, DFT has been extensively used to model materials ranging from superconductors [11], energy storage [16], to nanotechnology [3], among many others, playing a critical role in the development of new technologies and discovery of new functional materials.

As the DFT method has become a cornerstone in the field of material discovery, numerous computational packages have been developed and released over the past decades. Notable examples include VASP [7], Quantum Espresso [4], and SIESTA [15]. Despite their widespread use, the advent of artificial intelligence has shifted research interest toward integrating AI techniques to enhance DFT. However, current DFT packages face several challenges in incorporating neural networks and other machine learning methods. Firstly, the optimization of DFT relies on the self-consistent field (SCF) algorithm, which involves iterative eigendecomposition and Direct Inversion in the Iterative Subspace (DIIS) [13] algorithms, making it difficult to achieve differentiability. Secondly, these packages are typically written in languages and frameworks that are not easily compatible with modern deep learning frameworks, further complicating their integration with AI techniques.

To address the need for enhancing the DFT method with machine learning, we have developed a novel JAX-based differentiable density functional theory framework, named *Jrystal*, for crystalline solid-state material modeling and discovery. This package offers several unique features that distinguish it from existing quantum chemistry packages:

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

- **Differentiability**: By leveraging the JAX library [1], *Jrystal* enables automatic differentiation, facilitating the integration of neural networks and other machine learning techniques into the DFT workflow. This makes it convenient to calculate gradients for all physical quantities.
- **Direct Optimization**: *Jrystal* adopts a direct optimization approach for solving DFT with plane wave bases. Unlike the traditional SCF optimization method, our framework defines the objective function as the total energy of the system, or the Gibbs free energy for finite temperature scenarios. We parameterize the orthogonal wave functions and minimize these parameters using gradient-based optimization algorithms such as Adam [8].
- **Functional Programming**: Inherited from JAX, our framework is developed in a functional programming style. It provides various pure functions to construct the entire computation process. All APIs are designed as pure functions, ensuring no side effects. This approach makes reasoning about code behavior easier and improves extensibility.

2 Method

Total Energy Minimization Density Functional Theory (DFT) [9] is a quantum mechanical method used to calculate the electronic structure of materials by transforming the complex many-electron problem into a system of non-interacting electrons, governed by effective potentials derived from electron density. Traditionally, the DFT equations are solved using the Self-Consistent Field (SCF) method, an iterative algorithm that can be challenging to differentiate. Instead, we adopt a total energy minimization approach for this problem. The total energy is defined as a functional of the wave functions $\{\psi_i\}$ and orbital occupation numbers $\{f_i\}$:

$$E[\{\psi_i\}, \{f_i\}] = T_s[\{\psi_i\}, \{f_i\}] + E_{\text{ext}}[n] + E_{\text{Hxc}}[n] + E_{\text{nuc}}. \tag{1}$$

In this equation, T_s is the kinetic energy, which is defined by $T_s = -\frac{1}{2}\sum_i f_i \int \psi_i^*(\nabla^2\psi_i)d\boldsymbol{r}$. The external potential energy E_{ext} and Hartree-exchange-correlation energy $E_{\text{Hxc}}[n]$ are given by $E_{\text{ext}} = \int v_{\text{ext}}(\boldsymbol{r})n(\boldsymbol{r})d\boldsymbol{r}$ and $E_{\text{Hxc}} = \frac{1}{2}\int \frac{n(\boldsymbol{r})n(\boldsymbol{r}')}{|\boldsymbol{r}-\boldsymbol{r}'|}d\boldsymbol{r}\boldsymbol{r}' + E_{\text{xc}}$, where E_{xc} is the exchange-correlation energy. Both E_{ext} and E_{Hxc} depend on the electron density $n(\boldsymbol{r})$ which is constructed from the wave functions and occupation numbers as $n = \sum_i f_i|\psi_i|^2$.

Parameterize Wave Functions For a periodic system, plane wave basis is often used, which is given by

$$\psi_{i\boldsymbol{k}}(\boldsymbol{r}) = \frac{1}{\sqrt{\Omega}}\sum_{\boldsymbol{G}} c_{i\boldsymbol{G}}(\boldsymbol{k}) \exp\left(\mathrm{i}(\boldsymbol{k}+\boldsymbol{G})\boldsymbol{r}\right). \tag{2}$$

In this expression, the wave function is indexed by both i and $\boldsymbol{k}$. $\boldsymbol{k}$ is a vector in the Brillouin zone, and $\boldsymbol{G}$ is a reciprocal lattice vector and Ω is the volume of the unit cell. The coefficients $c_{i\boldsymbol{G}}(\boldsymbol{k})$ correspond to the plane wave components and are indexed by the band index i, the k-point $\boldsymbol{k}$, and the reciprocal lattice vector $\boldsymbol{G}$. These coefficients are subject to an orthonormality constraint,

$$\sum_{\boldsymbol{G}} c^\dagger_{i\boldsymbol{G}}(\boldsymbol{k})c_{j\boldsymbol{G}}(\boldsymbol{k}) = \delta_{ij}. \tag{3}$$

This can be achieved using a QR decomposition, which effectively transforms the problem into a more computationally efficient form. For a given wave vector $\boldsymbol{k}$, we can represent the coefficients as a matrix denoted as $\boldsymbol{C}(\boldsymbol{k})$, which possesses orthogonal columns. The reparameterization of this orthogonal coefficients can be expressed as follows:

$$\boldsymbol{C}(\boldsymbol{k}) = \mathsf{QR}(\boldsymbol{X}_{\boldsymbol{k}}) \tag{4}$$

where $\boldsymbol{X}_{\boldsymbol{k}}$ denotes the set of variational parameters that are subject to optimization. This method ensures that the orthogonality of the coefficients is maintained during the optimization process.

Smearing Scheme for Finite-Temperature To involve the smearing scheme for direct optimization at finite temperature, it is a common practice to introduce an entropy term into the objective function to minimize the free energy [10, 14, 2, 6]:

$$E - TS \tag{5}$$

where T is an electronic temperature, and S is the entropy associated with the occupation numbers. In the case of the Fermi-Dirac distribution, the entropy is defined as,

$$S_{\text{FD}} = -\sum_{i,\boldsymbol{k}} f_{i\boldsymbol{k}}\log f_{i\boldsymbol{k}} + (1-f_{i\boldsymbol{k}})\log(1-f_{i\boldsymbol{k}}). \tag{6}$$

Computational Graph A differentiable computational graph of our method is presented in Fig. 1.

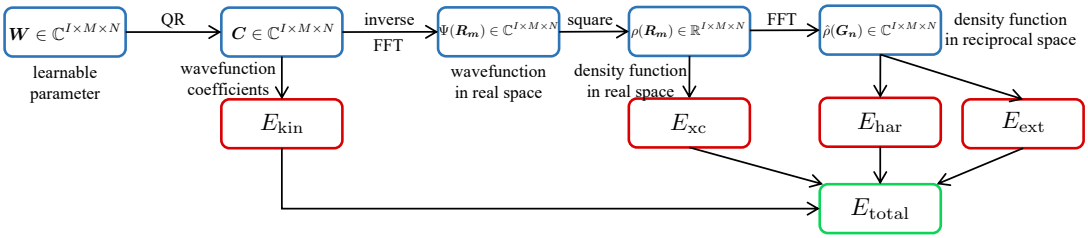

Figure 1: The computational graph shows the methodology for total energy minimization. Within this graph, arrows symbolize the sequence of forward computations. Every operation within this computational graph is differentiable, facilitating the gradient backpropagation.

3 The Framework

The *Jrystal* framework is built on JAX [1], a Python library that facilitates high-performance numerical computing and machine learning through automatic differentiation and GPU/TPU acceleration. Leveraging JAX, *Jrystal* provides user-friendly functions for performing computations related to atomic-scale material modeling and simulations. Below, we present examples demonstrating the novel features of our package.

Functional Programming Unlike other object-oriented programming packages, *Jrystal* defines plane-wave functions in a functional programming manner. This means there are no stateful parameters within the plane-wave object. Once the object is initiated, no internal parameter can be changed. The wave function can be created as follows:

```
>>> import jrystal as jr
# create a plane wave object with fft mesh 64*64*64 and 12 bands
>>> pw = jr.PlaneWave(num_bands=12, grid_sizes=[64, 64, 64])
# intialize parameter with a random key
>>> params = pw.init(key)
>>> wave_grid = pw.wave_grid(params, crystal)
```

In this example, `pw` is an immutable object. The `params` is the optimizable parameter, which is a PyTree object containing all the optimizable parameters of `PlaneWave`, including the including the parameters $\boldsymbol{X_k}$ and occupation number $\{f_{i\boldsymbol{k}}\}$. `crystal` is a dataclass that stores the crystal parameters. To obtain the wave function value evaluated at the mesh grid, you simply call the member function `wave_grid`, which is a pure function with no side effects.

Various APIs for DFT *Jrystal* provides various APIs for calculating energy functionals and other physical quantities, greatly facilitating DFT calculations. For example, the kinetic energy and total energy can be calculated as follows:

```
>>> kinetic_energy = jr.energy.kinetic(wave_grid, crystal)
>>> total_energy = jr.energy.total(wave_grid, crystal)
```

Another example is the calculation of the band structure, which is defined as the eigenvalues of the Hamiltonian matrix. This can be calculated in *Jrystal*:

```
>>> from jax.numpy.linalg import eigvalsh
>>> hamiltonian = jr.hamiltonian(wave_grid, crystal, ...)
>>> eigenvalues = eigvalsh(hamiltonian)
```

Automatic Differentiation Inherited from JAX, *Jrystal* can easily obtain all-order gradients of the functionals. A good example is calculating the forces in quantum systems. Thanks to JAX's automatic differentiation mechanism, forces can be easily obtained using `jax.grad`. The following example demonstrates how to compute the forces on the nuclei:

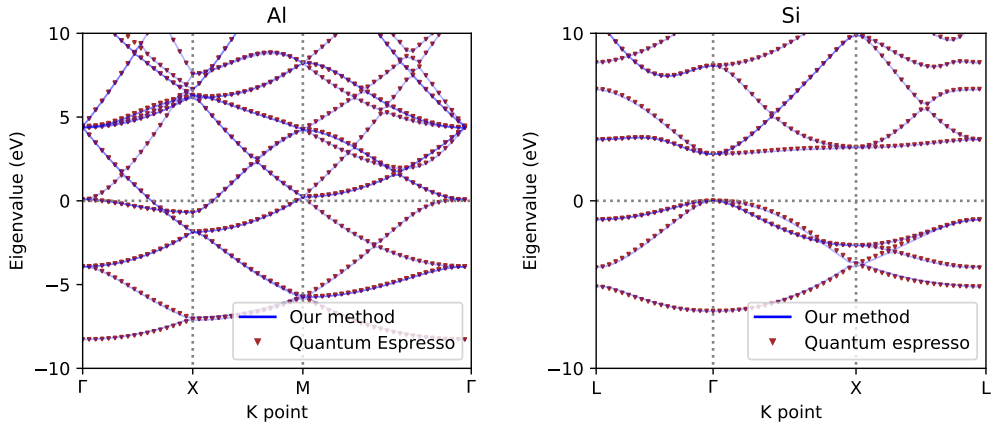

Figure 2: Comparison of the electronic band structures of Aluminum and Silicon calculated using the proposed method and Quantum Espresso. We use $3 \times 3 \times 3$ k-point mesh, cutoff energy of 100 Ha, smearing/temperature values of $T = 0.01Ha$ for both materials. As our method is an all-electron method, we have tuned the pseudo-potentials in our Quantum Espresso calculations to allow for an all-electron calculation. Quantum Espresso employs the conventional self-consistent field (SCF) Kohn-Sham DFT with LDA_X and a plane-wave basis in the pseudopotential projector-augmented wave formalism.

```
# define a total energy function w.r.t. nuclei's positions
>>> def total_energy(nuclei_positions):
>>>     wave_grid = pw.wave_grid(params, nuclei_positions, ...)
>>>     return jr.energy.total(wave_grid, crystal)
# calculate the force at nuclei_positions
>>> force = jax.grad(total_energy)(nuclei_positions)
```

These forces can be used in geometry optimization to update the positions of nuclei. With *Jrystal*, this calculation becomes very convenient and straightforward.

4 Results

We evaluate the effectiveness of our direct optimization approach in two tasks: band structure calculation (Figure 2) and geometry optimization (Figure 3).

In this test we compare our computed band structure of Aluminum and Silicon results to those computed the conventional self-consistent (SCF) method, as implemented in Quantum Espresso.[5] The resulting band structures are presented in Figure 2. The two methods yield similar energy band structures, indicating a strong agreement between the two approaches. This demonstrates the accuracy of the proposed method in reproducing the correct band structures and suggests that the occupation numbers and eigenfunctions are also well-aligned with the conventional SCF method.

In the second experiment, we examine the structure of a diamond-structured carbon crystal, which consists of two carbon atoms. One carbon atom is anchored at the origin, and we seek to ascertain the position of the second carbon atom as it moves along the plane highlighted in red, as depicted in Figure 3(d). The corresponding potential energy surface (PES) is computed and illustrated in Figure 3(a). The optimum is denoted by the star point, accurately representing the experimentally known diamond structure. Observations reveal the presence of multiple local optima and saddle points interspersed across the PES, underscoring the complexity of the energy landscapes typically found in atomic systems. Results show that both Yogi and Adam algorithms successfully reach the global optimum, while SGD lands at a local minimum (Figure 3(b)). In some cases, Yogi and SGD become stuck at a saddle point, whereas Adam finds the global optimum in the adjacent unit cell (Figure 3(c)). The local minimum found by Adam approaches a graphite-like atomic structure (Figure 3(f)).

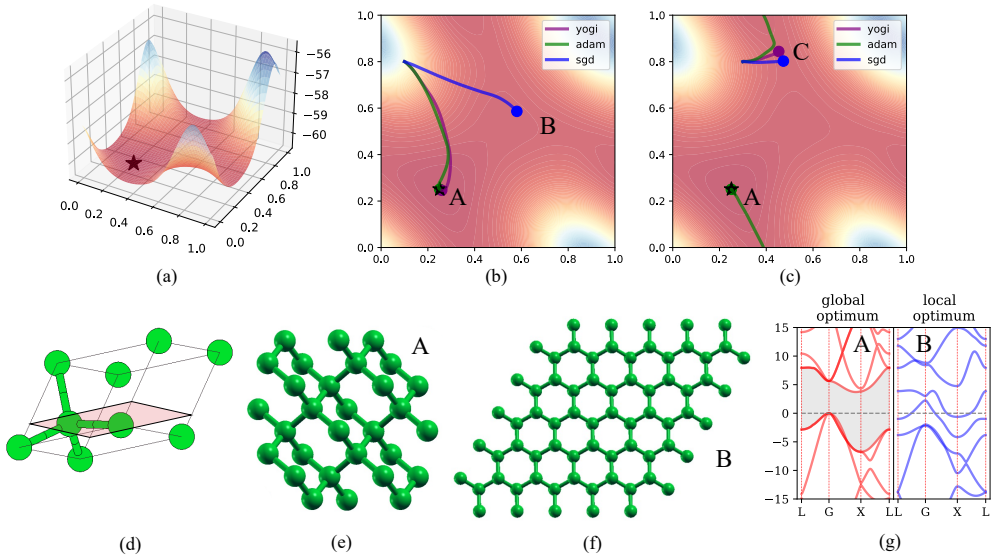

Figure 3: Illustration on the geometry optimization. The PES shown in (a-c) is computed by moving the atom in the centre of the unit-cell in (d) around the highlighted plane in a regular 50×50 grid. 'yogi', 'adam', and 'sgd' highlight optimization paths by the respective optimization method from their starting points as shown in (b, c). The global minimum is highlighted with a black star, and the geometry of the crystal at point A is given in (e), and consistent with the known structure of diamond. The geometry of the local minima at point B to which the sgd method converges to is presented in (f), which can be seen to approach the known structure of graphite. The different band structures of A and B are shown in (g).

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] Christoph Freysoldt, Sixten Boeck, and Jörg Neugebauer. Direct minimization technique for metals in density functional theory. *Physical Review B*, 79(24):241103, 2009.

[3] LJ Douglas Frink, AG Salinger, MP Sears, JD Weinhold, and AL Frischknecht. Numerical challenges in the application of density functional theory to biology and nanotechnology. *Journal of Physics: Condensed Matter*, 14(46):12167, 2002.

[4] Paolo Giannozzi, Oliviero Andreussi, Thomas Brumme, Oana Bunau, M Buongiorno Nardelli, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Matteo Cococcioni, et al. Advanced capabilities for materials modelling with quantum espresso. *Journal of physics: Condensed matter*, 29(46):465901, 2017.

[5] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.

[6] Xavier Gonze, Samare Rostami, and Christian Tantardini. Variational density functional perturbation theory for metals. *Phys. Rev. B*, 109:014317, Jan 2024.

[7] Jürgen Hafner. Ab-initio simulations of materials using vasp: Density-functional theory and beyond. *Journal of computational chemistry*, 29(13):2044–2078, 2008.

[8] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

[10] Nicola Marzari, David Vanderbilt, and Mike C Payne. Ensemble density-functional theory for ab initio molecular dynamics of metals and finite-temperature insulators. *Physical review letters*, 79(7):1337, 1997.

[11] Luiz Nunes de Oliveira, EKU Gross, and W Kohn. Density-functional theory for superconductors. *Physical review letters*, 60(23):2430, 1988.

[12] Aurora Pribram-Jones, David A Gross, and Kieron Burke. Dft: A theory full of holes? *Annual review of physical chemistry*, 66:283–304, 2015.

[13] Péter Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chemical Physics Letters*, 73(2):393–398, 1980.

[14] Álvaro Ruiz-Serrano and Chris-Kriton Skylaris. A variational method for density functional theory calculations on metallic systems with thousands of atoms. *The Journal of chemical physics*, 139(5), 2013.

[15] José M Soler, Emilio Artacho, Julian D Gale, Alberto García, Javier Junquera, Pablo Ordejón, and Daniel Sánchez-Portal. The siesta method for ab initio order-n materials simulation. *Journal of Physics: Condensed Matter*, 14(11):2745, 2002.

[16] Evan Walter Clark Spotte-Smith, Ronald L Kam, Daniel Barter, Xiaowei Xie, Tingzheng Hou, Shyam Dwaraknath, Samuel M Blau, and Kristin A Persson. Toward a mechanistic model of solid–electrolyte interphase formation and evolution in lithium-ion batteries. *ACS Energy Letters*, 7(4):1446–1453, 2022.